Assorted Python Recipes

Release

May 09, 2015

Contents

This *will* be a easy and searchable set of recipes for popular packages as well as standard library packages developed by myself and (hopefully) others.

The source can be found on GitHub, BitBucket, and Gitorious so pull requests will be accepted at all three places.

CHAPTER 1

pyserv: A simple bash function to use SimpleHTTPServer

Author Ian Cordasco

Origin http://www.coglib.com/~icordasc/blog/2012/11/one-liners.html

Tags bash, python, SimpleHTTPServer, screen,

1.1 The Final Code

```
pyserv() {
    if [[ -n $1 ]] ; then
        port="$1"
    else
        port="8000"
    fi

    old_dir="$(pwd)"

    if [[ -n $2 ]] ; then
        cd $2
    fi

    screen -dmS pyserv${port} python -m SimpleHTTPServer ${port}
    echo "http://localhost:${port}"
    echo "screen -r pyserv${port}"

    if ! [[ "$(pwd)" == $old_dir ]] ; then
        cd $old_dir
    fi
}
```

1.2 The Evolution

I have started to use a lot of tools that generate HTML for me and place them in pre-determined directories (e.g., coverage.py, sphinx, pelican). I can always use Firefox to navigate to those directories (`file:///home/sigma/sandbox/project/_build/html/index.html`) but that is annoying without a doubt. Sure I could bookmark that, but what if I move the project directory or change some settings? Wouldn't it be nice to just type a regular URL into the location bar? One solution is to just use python's `SimpleHTTPServer` module from the command-line like so

```
$ cd _build/html
$ python -m SimpleHTTPServer 8080
^C
$ cd -
```

But that takes over the terminal and sending it to the background introduces ugly interlacing of output. So I then started putting it in a screen session

```
$ cd _build/html
$ screen -dmS pyserver python -m SimpleHTTPServer 8080
$ cd -
```

Using `-dm` allows me to immediately place the screen session in the background instead of attaching it to the terminal. The `-S` option allows me to give it a name so I can do `screen -r pyserver` instead of `screen -ls; screen -r pidnumber`. But constantly having to type out the screen command was beginning to annoy me so I wrote a function

```
pyserv() {
    if [[ -n $1 ]] ; then
        port="$1"
    else
        port="8000"
    fi

    screen -dmS pyserv${port} python -m SimpleHTTPServer ${port}
    echo "http://localhost:${port}"
    echo "screen -r pyserv${port}"
}
```

This allowed me to not even need to set up a port by providing a default and echoed the url and screen command to the terminal for me. I could then just type

```
$ cd _build/html ; pyserv ; cd -
```

But constantly having to type `cd` was beginning to annoy me as well and this resulted in the final iteration of this function.

```
pyserv() {
    if [[ -n $1 ]] ; then
        port="$1"
    else
        port="8000"
    fi

    old_dir="$(pwd)"

    if [[ -n $2 ]] ; then
        cd $2
    fi

    screen -dmS pyserv${port} python -m SimpleHTTPServer ${port}
    echo "http://localhost:${port}"
    echo "screen -r pyserv${port}"

    if ! [[ "$(pwd)" == $old_dir ]] ; then
        cd $old_dir
    fi
}
```

It has all the simplicity of the original function with more functionality:

- You can simply call `pyserv` in a directory and it will work
- It gives you the URL and screen alias.
- It will now change directories for you and serve the content from there, while placing you back in your original directory

A couple simple invocations are

```
$ pyserv 8080 _build/html
$ pyserv 9090 htmlcov/
```

By placing this in your `.bashrc`, `.bash_profile` or any other bash related file, (and then sourcing it `. (filename)`) you can have immediate access to this functionality.

CHAPTER 2

How to Add Logging to a File When Using Requests

author Ian Cordasco

tags requests, logging, 1.x

In the latest versions of [Requests](https://github.com/kennethreitz/requests) the ability to configure sessions was removed and with it the ability to set a `verbose` option with a file-like object. Recently, a user stepped into `#python-requests` on Freenode and asked how to restore the behavior from before the refactor.

If one goes back in time (`git checkout 9dce7861134c60596b91dfa8eda2ff66f5c5d5e5`), you will see (using `grep`) that the call took place in `models.py` and simply formatted a string using `datetime.now().isformat()`, the request method and the URL. We no longer have that available to us, so the next best thing is to use `urllib3`'s built in logging. If you wanted, you could make it log to stderr by doing

```
import requests
requests.packages.urllib3.add_stderr_logger()
```

But then you'll only have the logged output printed to your terminal and that is not what the user wanted. Another way of doing this is to use the logging module yourself

```
import requests
import logging

logger = logging.getLogger('requests.packages.urllib3')
fh = logging.FileHandler()
# ...
logger.addHandler(fh, level=logging.DEBUG)
```

2.1 Why does this work?

`urllib3` uses it's namespace to register its own logger with the logging module (roughly speaking). In this instance, `urllib3` isn't a standalone package but part of `requests`, so its proper namespace is `requests.packages.urllib3`. Using that information, you can get the `logging.Logger` instance and configure it to your needs and desires. Note, however, that this will give you more detail than the old `verbose` configuration option did. As such, choose your level as you see fit.

CHAPTER 3

Contributing

If you want to make a contribution to these docs, fork the project on any of the three places it is available (GitHub, BitBucket or Gitorious).

- Please ensure that what you are contributing is now already protected under copyright by your employer and that you have full legal ability to contribute the code.
- Please provide an explanation of situations where the code is useful and tag your post with relevant words (to make it easily searchable).
- All source code longer than 25 lines (unless it is incomplete) should be in a separate file and should be included in the text using the `.. literalinclude::` Sphinx directive. The file should reside in the `source/` directory. For example:

```
# file.py
import requests

# do magic with requests
r = requests.get('http://httpbin.org/redirect/5',
                 allow_redirects=False)
# etc
```

```
Here is my awesome ``file.py`` which does ...

.. literalinclude:: source/file.py
    :language: python

etc.
```

- All complete code files should pass flake8.
- All links should have the URL at the bottom of the document to make reading the plain reStructuredText easier on future editors. Think of links as footnotes.
- Add yourself to `AUTHORS.rst` if you're contributing your own recipe.
- Add yourself to `EDITORS.rst` if you're contributing edits to other people's recipes.
- The recipes are not strictly restricted to python, but should be related to python.

CHAPTER 4

Contributors

4.1 Authors

- Ian Cordasco <graffatcolmingov@gmail.com>

4.2 Editors

If you want to make a contribution to these docs, fork the project on any of the three places it is available (GitHub, BitBucket or Gitorious).

- Please ensure that what you are contributing is now already protected under copyright by your employer and that you have full legal ability to contribute the code.
- Please provide an explanation of situations where the code is useful and tag your post with relevant words (to make it easily searchable).
- All source code longer than 25 lines (unless it is incomplete) should be in a separate file and should be included in the text using the `.. literalinclude::` Sphinx directive. The file should reside in the `source/` directory. For example:

```
# file.py
import requests

# do magic with requests
r = requests.get('http://httpbin.org/redirect/5',
                 allow_redirects=False)
# etc
```

```
Here is my awesome ``file.py`` which does ...

.. literalinclude:: source/file.py
    :language: python

etc.
```

- All complete code files should pass flake8.
- All links should have the URL at the bottom of the document to make reading the plain reStructuredText easier on future editors. Think of links as footnotes.
- Add yourself to `AUTHORS.rst` if you're contributing your own recipe.
- Add yourself to `EDITORS.rst` if you're contributing edits to other people's recipes.

- The recipes are not strictly restricted to python, but should be related to python.
- Ian Cordasco <graffatcolmingov@gmail.com>